# Cast-as-Intended Verifiability in Blockchain-based Electronic Voting

Matile, Raphael ; Rodrigues, Bruno ; John Scheid, Eder ; Stiller, Burkhard

# CaIV: Cast-as-Intended Verifiability in Blockchain-based Voting

Raphael Matile, Bruno Rodrigues, Eder Scheid, Burkhard Stiller
Communication Systems Group CSG, Department of Informatics IfI, University of Zurich UZH
Binzmühlestrasse 14, CH-8050 Zürich
E-mail: raphael.matile@uzh.ch [rodrigues,scheid,stiller]@ifi.uzh.ch

***Abstract*—Democracy in the digital age has attracted a lot of public attention in recent years. However, bringing the human right of secrecy in voting to electronic systems is difficult. Properties, such as the possibility of verifying universally that any vote counted was indeed carrying the decision made by a voter, are often conflicting and a trade-off must be found. This paper proposes a blockchain-based electronic voting system providing explicitly cast-as-intended verifiability. By using a non-interactive zero-knowledge proof of knowledge any voter can verify that his or her encrypted vote represents the decision voted for while maintaining at the same time the secrecy of the ballot. In addition, any required cryptographic material can be generated in linear time with respect to the number of voters, making the system suitable for large scale elections, thus scalable.**

## I. Introduction

Participating in voting and elections is a phenomenon which determines the fundamentals of democracy since centuries. However, different means have been used to cast a vote. From a spoken word, raising hands, to papyrus or paper, a multitude of representations for a ballot are known and used. In the last two decades, also electronic ways for casting a vote have been evaluated and introduced in electoral processes. From 1996 to 2007, only a total of 136 elections used a form of remote electronic voting [1], besides the uncounted ones of association internal elections of presidents, officers, or members of a board. Since then, advances in technology have lead to further trials and binding.

In Europe, efforts for introducing electronic voting have succeeded in a few countries. For example, Estonian citizens can submit votes over the Internet based on their previously established usage of a digital identity called SmartID [2]. Estonia has seen legally-binding electronic voting since 2005 [3] and also Switzerland has been experimenting with electronic voting channels since the beginning of the 21st century [4].

Although voices raised that electronic voting increases voter turnout, others state that only insignificant changes have been observed. With electronic voting channels, the key question of trust arises as soon as intangible ballots are submitted. Democracies with their checks and balances may already provide trusted means of voting. However, the recent presidential election of the United States of America in 2016 has shown that in long living democracies trust concerns in elections occur [5]. In other countries, transparent electronic voting systems are even considered the only means of conducting trustful elections [6].

978-1-7281-1328-9/19/$31.00 

Instead of relying on the honesty of an administering governmental entity, decentralized structures with independent implementations of a particular process cannot only distribute the risk of being attacked successfully, but might also increase the immediacy of a voter's participation in elections. With the publication of Bitcoin [7] in late 2008, decentralized ledgers have shown great potential for different IT systems and applications. Attempts to implement electronic voting on top of such a decentralized structure have been seen in the past [8], [9]. Although properties for evaluating pivotal requirements exists, few approaches take them into consideration. In particular, coercion-resistance and receipt-freeness, ballot secrecy, and verifiability of each submitted vote must be taken into account [10], [11]. With a country's federal structure, one fundamental requirement for decentralized electronic voting is already recognized: Distributing parts of the voting process to multiple authorities. Thus, establishing a similar structure for an electronic voting system seems to be the only logical consequence.

Once a vote is cast, it must be ensured that it represents the actual choice made by a voter, *i.e.*, whether the vote was cast as intended. In contrast to a paper ballot voting, electronic devices may alter votes invisibly to the end-user or they even might execute a different protocol from the one expected to be in use. This characteristic of all election systems - paper-based and electronic ones - is commonly referred to as "cast-as-intended verifiability" [10]. An electronic voting system providing recorded-as-cast and tallied-as-recorded verifiability combined with cast-as-intended verifiability, is said to be end-to-end verifiable [10].

Current implementations of electronic voting systems providing cast-as-intended verifiability commonly require multiple interactive steps by users [9], [8], [12]. Although maintaining state in a decentralized network is possible (*e.g.*, by using a distributed hash table), decentralized ledgers act in terms of transactions. Thus, this work designs CaIV, a blockchain-based electronic voting system using the universal cast-as-intended verifiability [13], represented in a self-contained manner within a single transaction per vote.

Therefore, after a brief review of related work in Section II. Section IV introduces the proposed design. Further, discussion and future work are presented in Section V.

## II. Related Work

The Swiss Post distributes within Switzerland its eVoting solution with an implementation based on return codes [14]. Votes can be submitted by prior authorization to the voting server. Based on the ballot received, a set of return codes is generated and transmitted to the voter. These return codes must be verified by the voter before submitting a confirmation message to the voting server again. Subsequently, a finalization code is generated, stored, and sent back to the voter. A vote is considered to be successfully received, if the finalization code matches an equivalent on the printed voting card [9].

Likewise, the canton of Geneva in Switzerland uses a similar end-user process, but a different implementation. Before votes are decrypted and tallied, they are mixed and partially decrypted by a set of voting authorities. These partially decrypted values are then retrieved by an election administrator and converted back to plain-text values. Finally, these values are summed up to calculate the final tally [8].

A feasibility study of a boardroom voting solution based on a Smart Contract (SC) running on the Ethereum blockchain is presented in [15]. A a Public Bulletin Board (PBB) stores all the election information. Votes are encrypted prior to sending them to the Ethereum network. Further, a 1-out-of-2 zero-knowledge proof ensures that the encrypted vote contains either a zero or one vote. The entire voting process includes a setup, sign-up, commit (optional), vote, and tally phase, of which the sign-up, commit, and vote phase have to be performed by the voter. Participants have to announce their private voting key during the sign-up phase and then commit to their voting option chosen by publishing a hash of their encrypted vote. Voters can send this vote and the zero-knowledge proof in a transaction to the SC. The major drawback is that the self-tallying voting protocol allows the last voter to abort the entire election.

[16] proposes an electronic voting solutions based on ring signatures on the Bitcoin blockchain. After a two-step registration phase, the voter is signed up for participating in the election. During voting each participant has to retrieve all public keys of all other voters, which he will use in combination with its own private key in order to sign the election option chosen. A commitment is sent to the Bitcoin address of the election authority. To tally, the voting authorities collect all transactions retrieved and verify their signature. If these are valid, the corresponding counter for a particular candidate option is increased. The paper claims to have fulfilled individual verifiability, however, no formal proof is provided.

Agora [17] has performed a trial voting on their blockchain-based implementation of an electronic voting system for the presidential elections in Sierra Leone. As registered election observers, members of Agora were allowed to manually register votes in 280 polling stations [17], [18].

## III. Cast-as-Intended Verifiability (CaIV)

CaIV allow voters to take political action in an uncontrolled decentralized environment (*i.e.,* the Internet). Therefore, it is best categorized as *eVoting* within the classification of New Voting Technology introduced by [19]. As such, the system requires at least two bodies operating different components during an election: *Voting authorities*, which operate a component to manage, store, tally votes, and code executed on the *Voters*' end-clients, such as a computer.

Maintaining durable connections between nodes in a distributed system is not straightforward: Nodes can disconnect from the network at any time due to many reasons. Therefore, using a protocol for the cast-as-intended verifiability, which requires only a single interaction, is preferable. Indeed, the protocol outlined in [13] allows to submit a vote in a single step and is considered a good fit for this problem.

In eVoting systems, a variety of data is usually made available on a public accessible storage: Not only public-private key-pairs used for encrypting votes, but also information required for providing verifiability or proofs for the validity of the final tally. According to [20], [10], such a Public Bulletin Board (PBB) provides the following properties:

1) It is an append-only data structure, *i.e.,* information cannot be modified or altered.
2) It is public in the sense of being searchable by anyone.
3) It is consistent in its view for anyone accessing its information.

However, Smart Contracts (SCs) running on Ethereum pose limitations to perform cryptographic operations: *(i)* Data types are only capable of storing a limited number of bytes and *(ii)* they are not supporting an implementation for big integers. Thus, eVoting based on the homomorphic ElGamal cryptosystem operating on finite cyclic groups of integers is, thus, infeasible as it relies on a large enough prime numbers as its modulus. However, a custom tailored blockchain as designed in this work, can take into account the verification of proofs defined in cyclic fields of integers.

## IV. CaIV Design

CaIV is structured into four main modules. The module **`crypto_rs`** provides the arithmetic foundation on which the additive homomorphic ElGamal cryptosystem is built. In addition, the implementations of non-interactive zero-knowledge proofs are contained. The **`generator_rs`** module creates all necessary cryptographic parameters, such as the private and public UCIV information and the election public-private key-pair. **`node_rs`** provides the PBB implemented as a blockchain, whereas **`client_rs`** provides all functionality to administrate and participate in the election.

### A. The crypto-rs Module

**`crypto_rs`** module provides an implementation of modular arithmetics in cyclic groups, an ElGamal ciphertext and a corresponding key-pair, and the NIZKPK (Non-Interactive Zero Knowledge Proofs) to verify the ciphertexts.

Based on `ModInt`, a data structure providing modular arithmetic, the additive variant of the homomorphic ElGamal ciphertext is specified by $G, H$ and the random number $r$ used during encryption. The encryption, decryption, and homomorphic addition are implemented as follows: Define the

message space of all valid plain-text votes to be in the cyclic subgroup $G$ of order $q$ of $(\mathbb{Z}_p)^*$, with $q$ being co-prime to $p$ and $g$ being the generator of $G$. Continue as follows:

1) Create a *private key* by selecting a random number $x$ from the uniformly distributed set $\{1, ..., q-1\}$ and keep $x$ secret.
2) Create the corresponding *public key*, by calculating $h = g^x$. Make the set $(G, q, g, h)$ public.
3) Encrypt a message $m \in \mathbb{Z}_p$ using $r \in_{\text{uniform}} \mathbb{Z}_p$ with the public key $h$ by calculating the shared secret $s = h^r = (g^x)^r = g^{xr}$. Then, the resulting ciphertext is defined as $E(G, H) = (g^r, g^m \cdot s)$.
4) Decrypt a ciphertext $E(G, H)$, by recalculating the secret $s = G^x = (g^r)^x = g^{rx}$ and $g^m = H \cdot (s^{-1}) = g^m \cdot h^r \cdot (g^{xr})^{-1} = g^m \cdot g^{xr} \cdot g^{-xr}$ with $s^{-1}$ being the modular multiplicative inverse of $s$. Solve the discrete logarithm in order to obtain $m$.

Having obtained two ElGamal ciphertexts $E(m_1)$ and $E(m_2)$ the homomorphic addition can be performed:

$$\begin{aligned} E(m_1) \cdot E(m_2) &= E(G_1, H_1) \cdot E(G_2, H_2) \\ &= E(g^{r_1}, g^{m_1} \cdot h^{r_1}) \cdot E(g^{r_2}, g^{m_2} \cdot h^{r_2}) \\ &= E(g^{r_1+r_2}, g^{m_1+m_2} \cdot h^{r_1+r_2}) \\ &= E(m_1 + m_2) \end{aligned}$$

In order to transform the ElGamal range proof to its non-interactive form, the Fiat-Shamir heuristic [21] is used. Similarly to the range proof, [13] defines the cast-as-intended verification proof in its non-interactive form.

### B. The generator-rs Module

The **`generator_rs`** is able to generate all cryptographic material, such as the private and public UCIV information and the election key-pair. However, creating a new ElGamal key-pair is not yet cryptographically safe: As of today, there is no safe prime generator available for the `BigInt` abstraction applied. In other words, the prime modulus $p$, its co-prime $q$, the private key $x$ are currently hard-coded. Thus, `generator_rs` is a best-effort solution for being able to process the entire voting procedure. However, this must be replaced once an implementation may become available.

### C. The node-rs Module

The implementation of the blockchain acting as PBB is provided in **`node_rs`**. Thus, it requires `crypto_rs` as dependency, providing it with the data structures on which transactions and blocks are built upon. Figure 1 shows its components: ❶ Blockchain nodes communicate using the `Node RPC interface`. ❷ Votes are submitted to a dedicated `Client RPC interface`. Besides two threads listening for incoming connections, a thread pool ❸ also runs another thread, signing blocks if the node is a leader or co-leader. ❹ The `Clique Protocol Handler` operates on the Proof-of-Authority level, where it handles incoming messages and corresponding responses, creates transactions and blocks, and decides whether the node is a leader or co-leader for the current epoch. In addition, it holds a transaction buffer ❺ and its own instance of the actual blockchain ❻.

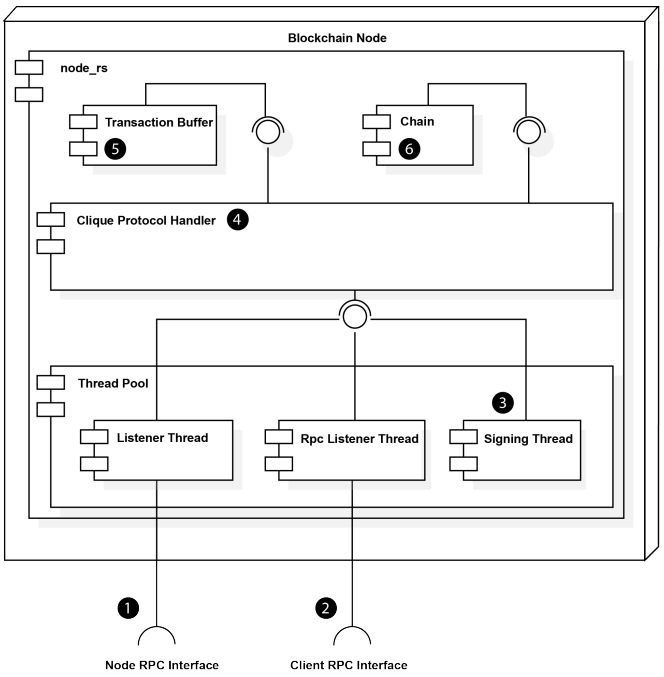


Fig. 1: CaIV architecture. ❶ ❷ denote the external communication interfaces. ❸ depicts a thread signing blocks. ❹ is the actual proof-of-authority protocol handler with its transaction buffer ❺ and the blockchain ❻.

The `configuration` module contains utilities to read all configuration required into a representation, which can be used as a genesis block for the blockchain. Among a version flag, it contains the block period for the Clique protocol [22], the number of blocks each node in the network is allowed to sign consecutively, the election public key $pk_e$, and the public UCIV information $uciv_p$ required to verify the cast-as-intended-verification proof of any transaction. During the runtime instantiation of the Clique protocol, the genesis block is hashed and the resulting value used to determine whether a node in the network is based on the same configuration. If a configuration value is different, also the hash will become different and, thus, nodes will never agree on the same canonical chain. Therefore, nodes with a different genesis block hash are excluded from communications.

The data of the blockchain itself is stored on the heap. Instead of using a tree-based data structure to build up the chain, an adjacent matrix is created, containing on the $y$-axis all block identifiers and on the $x$-axis corresponding children identifiers of the block. This ensures $\mathcal{O}(1)$ cost when looking up a particular key.

### D. The client-rs Module

In the current design, all functionality for administrating elections, submitting votes, and obtaining a final tally is combined into the one client application **`client-rs`**.

### E. Vote Administration

Election authorities open and close the election by sending an `OpenVote` and `CloseVote` message to the bulletin board, respectively. Incoming vote transactions will only be counted towards the final tally, if the election is opened and ignored otherwise. To submit a vote, voters only need to be in possession of the election public key $pk_e$ and their associated private and public UCIV information pair $(uciv_s, uciv_p)^{vid}$. Voters can type either `yes` or `no` as answer to the voting question. This value is transformed in to its binary representation $1$ or $0$, respectively. This binary vote is further encrypted on the voter's client device, and the range proofs as well as the cast-as-intended verification proof are created. Before submitting the vote to the blockchain, both proofs are validated and the vote is submitted to a node of the voting network only if the proof validation is successful.

Once the election has been closed by voting authorities, the final tally can be calculated. By sending a `RequestTally` to a blockchain node, a traversal from the root to the end of its current canonical chain is initiated. Each block's vote transactions $T_V$ are homomorphically summed up. Beforehand, each vote transaction is validated by their associated proofs. In case a proof fails to evaluate successfully, the transaction is counted toward invalid votes. Once the entire canonical chain has been traversed, a response containing the amount of successful, invalid and total votes is returned. If no `CloseVote` transaction is observed during the traversal of the canonical chain, zero-values are returned for the above metrics.

## V. Discussion and Future Work

Although a secure electronic voting system is of absolute necessity, the integrity of the voting question itself is crucial as well. Tampering with its linguistic structure, such as negating or rephrasing it entirely, can affect the outcome of the election effectively. Thus, storing it on a distributed ledger can substantially reduce such a risk.

Besides the cast-as-intended verifiability, *recorded-as-cast verifiability* can be ensured as well too: If a voter obtains the identifier of the transaction submitted to the blockchain, it can be queried after submission. As such, the voter can verify the associated proofs for correctness. However, as transactions are not signed with a voter-dependent value the integrity verification of the vote on the blockchain becomes poor. Also, the transaction's integrity can be verified by computing the hash of the transaction before submitting it to the blockchain and then asserting for equality during verification.

*Counted-as-recorded verifiability* is not yet provided by the implementation proposed as this work primarily focused on providing *cast-as-intended* verifiability. The current voting protocol can be extended in the last step by (a) computing a non-interactive zero-knowledge proof of knowledge for the correct tabulation and (b) publishing it to the bulletin board.

*Individual verifiability* can be provided if the voter has a means of retrieving his/her vote from the blockchain and evaluating it for correctness using the associated proofs. *Universal verifiability* may be provided, if a tabulation proof of the final tally is published to the PBB. Due to these limitations, the implementation performed and run in a test set-up does not yet fulfil the end-to-end verifiability completely.

Although votes are locally encrypted on voters' end-user devices, election authorities are still able to decrypt individual votes, if they have a means of querying the blockchain for individual transactions. As this is suggested for providing *recorded-as-cast* verifiability, it will be hard to refuse this access to election authorities, especially as they provide the infrastructure running the blockchain. One can overcome this limitation by establishing multiparty computation as suggested in [13]. Also, [13] states additionally that the suggested voting protocol on which this work is based on, trusts the election authorities for privacy reasons, a partially debatable assumption.

Thus, the CaIV shows that by storing the non-interactive zero-knowledge proof of knowledge [13] on the blockchain, cast-as-intended verifiability is guaranteed while still maintaining a limited notion of privacy. In addition, tabulation of the final tally is performed directly on the blockchain, distributing the trust for its integrity among multiple authorities. However, the opposing properties of verifiability and privacy also revealed challenges for providing recorded-as-cast and counted-as-recorded verifiability in a distributed setup: Ballot privacy can be reduced by homomorphically tabulating directly on the blockchain. With respect to Swiss national elections and voting, extensions to the current proof-of-concept implementation must be provided in order to fulfil the fundamental property of end-to-end verifiability.

Additional implementation-specific aspects require attention prior to performing a nationwide election. For example, performing the homomorphic addition on the blockchain leaks information about the randomness of the ciphertext in a public manner, which is achieved by reducing the solution space that an adversary needs to consider when trying to decrypt a vote. In other words, an adversary would need to explore less solutions to decrypt a ciphertext and correlate an id with a vote. Also, in a production-ready application incoming messages are to be signed with an asymmetric key-pair by each election authority. Thus, illegitimate requests attempted by non-authority entities can be filtered out. Finally, although empty votes can occur by voters who register to vote but never submit a decision to the voting network, voting for one or even multiple options shall be considered, which has to be extended for write in-names and candidates, too.

Although a blockchain can provide trust among different authorities, the cryptographic procedures to ensure a secure and integer vote are highly complex. Critics of eVoting systems argue that a large part of the electorate will not be able to understand the verifiability properties in detail. Thus, future work also needs to show whether a more human readable kind of verifiability can be constructed.

## ACKNOWLEDGEMENTS

The authors would like to thank Christian Killer for the many in-depth discussions and feedback for the final version.

## REFERENCES

[1] R. Krimmer, S. Triessnig, and M. Volkamer, " *E-Voting and Identity* "The Development of Remote E-Voting Around the World: A Review of Roads and Directions, A. Alkassar and M. Volkamer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1–15.

[2] State Electoral Office of Estonia, "General Framework of Electronic Voting and Implementation thereof at National Elections in Estonia," pp. 1–21, 2017. [Online]: https://bit.ly/2XTx7ZB

[3] D. Clarke and T. Martens, "E-Voting in Estonia," *Real-World Electronic Voting: Design, Analysis and Deployment*, pp. 129–141, 2016.

[4] Die Schweizerische Bundeskanzlei (2), "Vote Electronique - Chronik," 2018, Accessed: Dec. 12, 2018. [Online]: https://bit.ly/2UybprV

[5] M. R. Holman and J. C. Lay, "They See Dead People (Voting): Correcting Misperceptions about Voter Fraud in the 2016 U.S. Presidential Election," *Journal of Political Marketing*, Vol. 6, No. 2, pp. 1–38, 2018. [Online]: https://doi.org/10.1080/15377857.2018.1478656

[6] M. Hapsara, A. Imran, and T. Turner, " *Electronic Voting* "E-Voting in Developing Countries, R. Krimmer, M. Volkamer, J. Barrat, J. Benaloh, N. Goodman, P. Y. A. Ryan, and V. Teague, Eds. Cham: Springer International Publishing, 2017, pp. 36–55.

[7] S. Nakamoto, "Bitcoin: A Peer-To-Peer Electronic Cash System," 2008. [Online]: https://bit.ly/LjkXCv

[8] R. Haenni, R. E. Koenig, P. Locher, and E. Dubuis, "CHVote System Specification – Version 1.4.2," *IACR Cryptology ePrint Archive*, Vol. 2017, p. 325, 2018. [Online]: http://dblp.uni-trier.de/db/journals/iacr/iacr2017.html#HaenniKLD17

[9] D. Galindo, S. Guasch, and J. Puiggalí, " *E-Voting and Identity* "2015 Neuchâtel's Cast-as-Intended Verification Mechanism, R. Haenni, R. E. Koenig, and D. Wikström, Eds. Cham: Springer International Publishing, 2015, pp. 3–18.

[10] H. Jonker, S. Mauw, and J. Pang, "Privacy and Verifiability in Voting Systems," *Computer Science Review*, Vol. 10, pp. 1 – 30, 2013. [Online]: https://bit.ly/2PPATOe

[11] J. Benaloh, M. Bernhard, J. A. Halderman, R. L. Rivest, P. Y. A. Ryan, P. B. Stark, V. Teague, P. L. Vora, and D. S. Wallach, "Public Evidence from Secret Ballots," 2017. [Online]: http://arxiv.org/abs/1707.08619

[12] A. Brelle and T. Truderung, "Cast-as-Intended Mechanism with Return Codes Based on PETs," 2017. [Online]: http://arxiv.org/abs/1707.03632

[13] A. Escala, S. Guasch, J. Herranz, and P. Morillo, "Universal Cast-as-Intended Verifiability," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 9604 LNCS, pp. 233–250, 2016.

[14] SwissPost, "Swiss Posts e-voting Solution," 2019, Accessed: Mar. 03, 2019. [Online]: http://www.post.ch/evoting

[15] P. McCorry, S. F. Shahandashti, and F. Hao, " *Financial Cryptography and Data Security* "A Smart Contract for Boardroom Voting with Maximum Voter Privacy, A. Kiayias, Ed. Cham: Springer International Publishing, 2017, pp. 357–375.

[16] Y. Wu, "An E-voting System based on Blockchain and Ring Signature," 2017, Accessed: Mar. 12, 2019. [Online]: https://www.dgalindo.es/mscprojects/yifan.pdf

[17] Agora, "Agora: Bringing our Voting Systems into the 21st Century," p. 46, 2017, Accessed: Dec. 12, 2018. [Online]: https://bit.ly/2A863LR

[18] Agora, "Agora Official Statement Regarding Sierra Leone Election," https://bit.ly/2BvLqcx, 2018, Accessed: Oct. 10, 2018.

[19] OSCE/ODIHR, *Handbook For the Observation of New Voting Technologies*. OSCE Office for Democratic Institutions and Human Rights (ODIHR), 2013. [Online]: https://bit.ly/Ua6Q6T

[20] H. Jonker and J. Pang, "Bulletin Boards in Voting Systems: Modelling and measuring privacy," *Proceedings of the 2011 6th International Conference on Availability, Reliability and Security, ARES 2011*, pp. 294–300, 2011.

[21] A. Fiat and A. Shamir, "How to Prove Yourself: Practical solutions to Identification and Signature Problems," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 263 LNCS, pp. 186–194, 1987.

[22] P. Szilgyi, "Clique PoA protocol & Rinkeby PoA testnet," https://bit.ly/2J5DCVK, 2017, Accessed: Dec. 11, 2018.